Symbolic-Numeric Methods for Nonlinear Integro-Differential Modeling

François Boulier, Hélène Castel, Nathalie Corson, Valentina Lanza, François Lemaire, Adrien Poteaux, Alban Quadrat, Nathalie Verdière

Symbolic-Numeric Methods for Nonlinear Integro-Differential Modeling

François Boulier[1], Hélène Castel[3], Nathalie Corson[2], Valentina Lanza[2], François Lemaire[1], Adrien Poteaux[1], Alban Quadrat[1], and Nathalie Verdière[2]

[1] Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France
[2] Normandie Univ, France, UNIHAVRE, LMAH, FR CNRS 3335, ISCN, 76600 Le Havre, France
[3] INSERM, DC2N, Normandie Univ, UNIROUEN, 76000, Rouen, France

Abstract. This paper presents a proof of concept for symbolic and numeric methods dedicated to the parameter estimation problem for models formulated by means of nonlinear integro-differential equations (IDE). In particular, we address: the computation of the model input-output equation and the numerical integration of IDE systems.

1 Introduction

This paper is concerned by the problem of modeling phenomena by systems of nonlinear integro-differential equations (IDE). Motivations for IDE modeling are presented in [14]. In turn, this scientific question raises the two following problems: how to determine the identifiability property of such IDE models? how to estimate parameters from experimental data? We focus on a particular method, called the "input-output (IO) ideal" method, which is available in the nonlinear differential case. The idea of this method consists in computing an equation (called the "IO equation") which is a consequence of the model equations and only depends on the model inputs, outputs and parameters. In the nonlinear differential case, it is known since [27] that it can serve to decide the identifiability property of the model. It is known since [17] that it can also be used to determine a first guess of the parameters from the experimental data. This first guess may then be refined by means of a nonlinear fitting algorithm (of type Levenberg-Marquardt) which requires many different numerical integrations of the model.

Designing analogue theories and algorithms in the IDE case is almost a completely open problem in spite of many recent progresses on the algebraic properties of integro-differential algebras and their operator rings [36, 19, 20, 2–4, 33].

This article provides two contributions:

1. a symbolic method for computing an IO equation from a given nonlinear IDE model. This method is incomplete but it is likely to apply over an important class of models that are interesting for modelers;

2. an algorithm for the numerical integration of IDE systems, implemented within a new open source C library, endowed with a new MAPLE package called `Blineide`. The library does not seem to have any available equivalent. Our algorithm is an explicit Runge-Kutta method which is restricted to Butcher tableaux specifically designed in order to avoid solving integral equations at each step. In this paper, we provide three such tableaux.

The structure of the paper is as follows. Section 2 provides examples of IDE equations and the symbolic method for computing an IO equation from an IDE model. Section 3 describes our algorithm for the numerical integration of IDE. Section 4 describes its implementation.

2 An IDE Input-Output Equation

This section starts with a short presentation of the Volterra-Kostitzin model, which gives some insight on the point of introducing kernels in models. The second section presents an academic IDE model and explains, over an example, how to compute an IO equation. The last section contains a discussion on how algorithmic is the process illustrated by the example.

2.1 The Volterra-Kostitzin Model

As pointed out in [14], one of the simplest nonlinear integro-differential models studied in the literature is the Volterra-Kostitzin model [26, pages 66-69] (more recently revisited in [32, chapter 4]), which may be used for describing the evolution of a population, in a closed environment, intoxicated by its own metabolic products (other applications of the same model are considered in Kostitzin's book). It is an integro-differential equation since the unknown function $y(x)$ appears both differentiated and under some integral sign.

$$\dot{y}(x) = \varepsilon\, y(x) - \lambda\, y(x)^2 - \mu\, y(x) \int_{x-T}^{x} K(x-\xi)\, y(\xi)\, \mathrm{d}\xi\,. \tag{1}$$

The independent variable x is time. The dependent variable $y(x)$ is the population, varying with time. The symbols ε, λ, μ and T denote parameters. The *kernel* (or *nucleus*) $K(x,\xi) = K(x-\xi)$ is the *residual action function*. For instance, it could be very similar to a "survival function" in population dynamics [23, page 3]: a decreasing function, starting at $K(0) = 1$, equal to 0 outside the interval $[0, T]$. Then $K(x-\xi)$ would represent the "toxicity factor" of metabolic products which are the most toxic when produced, at $x = \xi$, become less toxic with the time, and have a negligible toxic effect at time $x = \xi + T$.

In the case of models presented by chemical reaction systems, similar kernels could arise from stochastic considerations. Indeed, if the molecularity (the number of reactants) of each reaction is one, then the statistical moments of the random variables which count molecules can be described by ODE [31]. However, if the molecularity of some reactions is greater than one, then the ODE system

for the statistical moments becomes infinite and is in general very difficult to approximate by a finite system. A natural idea would then consist in tabulating the density probability of the event under consideration and incorporate the tabulated curve as an integral kernel in some IDE model. See [24, sect. 3.6].

2.2 A Compartmental IDE Model

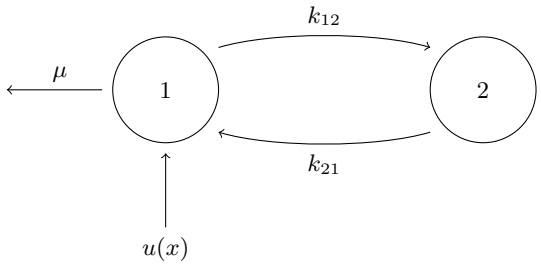

Fig. 1. A two-compartment model featuring three parameters.

The academic two-compartment model depicted in Figure 1 is a close variant of [40, (1), page 517] endowed with an input $u(x)$ and an IDE variant of the model studied in [14]. Compartment 1 represents the blood system and compartment 2 represents some organ. Both compartments are supposed to have unit volumes. The function $u(x)$, which has the dimension of a flow, represents a medical drug, injected in compartment 1. The drug diffuses between the two compartments, following linear laws: the proportionality constants are named k_{12} and k_{21}. In this paper, we assume that the drug exits compartment 1, following a law given by an integral term (this model is thus new), depending on a parameter μ (see the Volterra-Kostitzin model for a possible modeling argument). The state variables in this system are $z_1(x)$ and $z_2(x)$. They represent the concentrations of drug in each compartment. This information is sufficient to write the two first equations of the mathematical model (2). The last equation of (2) states that the output, denoted $y(x)$, is equal to $z_1(x)$. This means that only $z_1(x)$ is observed: some numerical data are available for $z_1(x)$ but not for $z_2(x)$. The problem addressed here then consists in estimating the three parameters k_{12}, k_{21} and μ from these data and the knowledge of $u(x)$.

In order to estimate the model parameters over such a model, the strategy of the "input-output ideal" method consists in computing from the model equations, an "input-output (IO) equation" featuring only the input $u(x)$, the output $y(x)$ and the unknown parameters. If the model were differential only, the computation of the IO equation, which is an elimination problem, could be handled by means of the elimination theory of differential algebra. See [27, 17] and references therein. The IO equation itself could be algebraically described as

the single differential polynomial of the regular differential chain are associated to some differential polynomial ideal of some differential polynomial ring. In the IDE case, there does not exist (yet) any integro-differential algebra theory, rich enough to enunciate such a precisely defined statement.

$$\begin{aligned}
\dot{z}_1(x) &= -k_{12}\, z_1(x) + k_{21}\, z_2(x) - \mu\, z_1(x) \underbrace{\int_0^x K(x-\xi)\, z_1(\xi)\, \mathrm{d}\xi}_{\text{integral term}} + u(x)\,, \\
\dot{z}_2(x) &= k_{12}\, z_1(x) - k_{21}\, z_2(x)\,, \\
y(x) &= z_1(x)\,.
\end{aligned} \tag{2}$$

2.3 A Work-around Strategy

It turns out that a work-around strategy is available for a wide class of IDE models. We present it over Model (2).

Renaming Integrals The idea consists in renaming the integral term using a new unknown function $F(x)$, yielding a polynomial differential model (3), and process this differential model by the classical IO ideal method.

$$\begin{aligned}
\dot{z}_1(x) &= -k_{12}\, z_1(x) + k_{21}\, z_2(x) - \mu\, z_1(x)\, F(x) + u(x)\,, \\
\dot{z}_2(x) &= k_{12}\, z_1(x) - k_{21}\, z_2(x)\,, \\
y(x) &= z_1(x)\,.
\end{aligned} \tag{3}$$

Model (3) can be viewed as a polynomial system of the differential polynomial ring $\mathscr{R} = \mathscr{F}\{z_1, z_2, y, u, F\}$, where $\mathscr{F} = \mathbb{Q}(k_{12}, k_{21}, \mu)$. As such, it generates a perfect (even a prime) differential ideal $\mathfrak{A}$. It is even a regular differential chain for $\mathfrak{A}$, with respect to some orderly ranking.

Eliminating State Variables By an elimination procedure (eliminating z_1 and z_2) one can compute a regular differential chain C_{io} such that $C_{\mathrm{io}} \cap \mathscr{F}\{y, F\}$ is a regular differential chain for the differential ideal $\mathfrak{A} \cap \mathscr{F}\{y, F\}$. The regular differential chain $C_{\mathrm{io}} \cap \mathscr{F}\{y, F\}$ is made of the following single differential polynomial

$$\begin{aligned}
D_{\mathrm{io}} = \ddot{y}(x) &+ \mu\, \dot{y}(x)\, F(x) + k_{12}\, \dot{y}(x) + k_{21}\, \dot{y}(x) - \dot{u}(x) + \mu\, y(x)\, \dot{F}(x) \\
&+ \mu\, k_{21}\, y(x)\, F(x) - k_{21}\, u(x)\,.
\end{aligned} \tag{4}$$

Integrating the IO Equation Applying an integration algorithm for differential fractions, one gets the following reformulation of (4)

$$\begin{aligned}
D_{\mathrm{io}} = {} & \mu\, k_{21}\, y(x)\, F(x) - k_{21}\, u(x) \\
& + \frac{\mathrm{d}}{\mathrm{d}x}\, \left((k_{12} + k_{21})\, y(x) + \mu\, y(x)\, F(x) - u(x) \right) + \frac{\mathrm{d}^2}{\mathrm{d}x^2}\, y(x)\,,
\end{aligned} \tag{5}$$

which can now easily be transformed into an integral equation (integrate twice between 0 and x and use the kernel $x-\xi$ (not to be confused with the kernel $K(x,\xi)$ present in model (3)) to encode double integrals by single ones — see [23, sec. 1.3.1]):

$$\begin{aligned} I_{\text{io}} = {} & \mu\, k_{21} \int_0^x (x-\xi)\, y(\xi)\, F(\xi)\, \mathrm{d}\xi - k_{21} \int_0^x (x-\xi)\, u(\xi)\, \mathrm{d}\xi \\ & + (k_{12}+k_{21}) \left(\int_0^x y(\xi)\, \mathrm{d}\xi - x\, y(0) \right) \\ & + \mu \left(\int_0^x y(\xi)\, F(\xi)\, \mathrm{d}\xi - x\, y(0)\, F(0) \right) \\ & - \left(\int_0^x u(\xi)\, \mathrm{d}\xi - x\, u(0) \right) + y(x) - y(0) - x\, \dot{y}(0)\,. \end{aligned} \tag{6}$$

Normalizing Integral Terms It is now time to replace $F(x)$ by its value (and $F(0)$ by 0). However, the expression under the integral sign involves an indeterminate (z_1) which is supposed to be eliminated. Since this expression is a differential polynomial, differential algebra tools can again be applied and we can replace z_1 by its normal form with respect to the regular differential chain C_{io}. Since this chain involves the equation $z_1 = y$, the normal form of z_1 is y and we actually replace $F(x)$ by

$$\int_0^x K(x-\xi)\ \mathrm{NF}(z_1, C_{\text{io}})(\xi)\, \mathrm{d}\xi = \int_0^x K(x-\xi)\, y(\xi)\, \mathrm{d}\xi\,.$$

The Resulting Equation After replacement, one eventually gets Equation (7), given in Fig. 2. In order to establish the global identifiability of model (3),

$$\begin{aligned} I_{\text{io}} = {} & \underbrace{\mu\, k_{21}}_{c_4} \underbrace{\left(\int_0^x (x-\xi)\, y(\xi) \int_0^\xi K(\xi-\tau)\, F(\tau)\, \mathrm{d}\tau\, \mathrm{d}\xi \right)}_{m_4} \\ & \underbrace{-\, k_{21}}_{c_3} \underbrace{\int_0^x (x-\xi)\, u(\xi)\, \mathrm{d}\xi}_{m_3} + \underbrace{(k_{12}+k_{21})}_{c_2} \underbrace{\left(\int_0^x y(\xi)\, \mathrm{d}\xi - x\, y(0) \right)}_{m_2} \\ & + \underbrace{\mu}_{c_1} \underbrace{\left(\int_0^x y(\xi) \int_0^\xi K(\xi-\tau)\, y(\tau)\, \mathrm{d}\tau\, \mathrm{d}\xi \right)}_{m_1} \\ & \underbrace{- \left(\int_0^x u(\xi)\, \mathrm{d}\xi - x\, u(0) \right) + y(x) - y(0) - x\, \dot{y}(0)}_{m_0}\,. \end{aligned} \tag{7}$$

Fig. 2. An IO equation for model 3.

the argument would be the following: Equation (7) is a linear combination

$c_1\, m_1 + \cdots + c_4\, m_4 = m_0$. In principle, the "monomials" m_i can be evaluated at different values of x over the experimental data, yielding a linear system whose unknowns are the blocks of parameters c_i. If the matrix of this linear system has full rank, the system can be solved, providing estimates for the blocks of parameters. Over this system, it is — in principle — easy to recover estimates of the model parameters k_{12}, k_{21}, μ from the estimates of the c_i. These questions are not addressed in this paper.

2.4 Discussion

By many aspects, the computation of Equation (7) from model (3) suggests algorithms for processing models presented by systems of IDE.

Renaming Integrals Indeed, it is always possible to rename many different integral terms by new unknown functions $F_i(x)$. The resulting model is a system of differential polynomials (more generally, of differential fractions) in the sense of differential algebra. If the initial IDE model is a dynamical system (i.e. is solved w.r.t. differentiated state variables z_i) then the resulting model defines a prime differential ideal and is a regular differential chain for this ideal, w.r.t. some orderly ranking.

Reference books for differential algebra are [35, 25]. Regular differential chains are generalizations of Ritt's characteristic sets. In the non differential context, regular chains provide an alternative to Gröbner bases for describing polynomial ideals and performing some ideal-theoretic constructs. In the differential context, the Gröbner bases theory does not generalize satisfactorily: regular differential chains and other close concepts are the only tools available for investigating properties of differential ideals. See [13] for a recent study of this concept.

Orderly rankings are defined in [25, I, 8, page 75]. The fact that the differential ideal defined by a dynamical system is prime follows from the fact that each equation of the regular differential chain is linear in its leading derivative, hence cannot be represented as the product of two differential polynomials with positive degree in this leading derivative.

Eliminating State Variables Eliminating the state variables can be achieved by means of a differential elimination algorithm [28, 10, 34, 11, 1], using some specific ranking, leading to some regular differential chain C_{io}. These elimination algorithms can be applied over any system of differential polynomials. They can also be applied over any system of differential fractions, by handling the numerators of the differential fractions as differential polynomial equations and the denominators as differential polynomial inequations (polynomials that are required to be nonzero). See the implementation of [7, RosenfeldGroebner]. Moreover, if the input model already is a regular differential chain w.r.t. some (orderly) ranking, it is possible to apply an improved elimination method [30, 12] which avoids splitting cases. Let us conclude this section by a few remarks:

– the state-of-the-art elimination algorithms do not try to minimize the number of times these unknown functions get differentiated, which might be problematic if the integral terms depend on (say) kernels which are not indefinitely differentiable. A similar issue arises in the case of PDE [42];
– if the integral terms satisfy some known differential algebraic relation, it is possible to enlarge the model equations with this relation before the elimination process.

Integrating the IO Equation For simplicity, let us assume that, among the many different differential polynomials occuring in the regular differential chain C_{io}, a single one (called the IO equation) is free of the state variables.

The integration algorithm [9] may be applied over the IO equation or over any equivalent differential fraction, obtained by dividing the equation by some other differential polynomial, such as the leading coefficient (the initial) of the IO polynomial. The result can then be converted into an IDE (such as (6)) by means of classical techniques. See [8] and [23, Formula (1.45)].

From a theoretical point of view, this integration step is not mandatory. In practice, it leads to formulas which are much more suitable for parameter estimation, as established in [29, 39].

Normalizing Integral Terms Substituting back the unknown functions $F_i(x)$ by the integral terms they represent does not raise any problem. The normalization of the expressions lying under the integral terms leads to a more subtle issue.

In general, an integral term involves, as sub-expressions, many different (e.g. in the case of nested integrals) differential fractions $[f_1, f_2, \ldots, f_r]$. The normal form algorithm presented in [6] can be applied over all these fractions, w.r.t. the whole regular differential chain C_{io}. These normal forms are themselves differential fractions $[\text{NF}_1, \text{NF}_2, \ldots, \text{NF}_r]$. Replacing each f by its normal form, one gets another formulation of the integral term, which is equivalent to the original one.

In full generality, the normal forms may themselves depend on unknown functions $F_i(x)$ and one may consider to iterate this substitution process. If the ranking w.r.t. which C_{io} is defined is not carefully chosen, the substitution process may transform an IO equation into a non-IO equation or (worse) may not terminate at all. A careful study of this issue is left for investigation in another paper.

The Resulting Equation If the resulting equation does not depend on the state variables at all, it is a candidate for an IO equation. However, in the absence of any sound integro-differential elimination theory, it is not clear that it is minimal. For similar reasons, if the resulting equation depends on state variables so that it is not an IO equation, it is not clear that no IO equation exists at all.

3 Numerical Integration of IDE Systems

According to [41], IDE are a particular case of delay differential equations (DDE) (continuous delay differential equations). However, though there exists numerical solvers for DDE with constant delays [37], there does not seem to be any widely available software for IDE. Within a whole section dedicated to DDE [21, sec. II.17], a single page is dedicated to the numerical integration problem of IDE in [21, page 351], which refers to [15] and sketches solutions in particular cases only. In this article, we focus on explicit Runge-Kutta methods. See [18] to a theoretical study of their application to the numerical integration of IDE. The relationship between these early works and our paper still needs some investigation.

3.1 The Method

We are concerned with the numerical integration of IDE of the form

$$\dot{y}(x) = f(x, y(x))\,, \tag{8}$$

over some integration interval $[x_0, x_{\mathrm{end}}]$. The independent variable x is real. The dependent variable y may actually be a vector of n functions of x. The function f may depend on inputs $u(x)$ and on integral terms of the form

$$\int_{\alpha(x)}^{\beta(x)} K(x,\xi)\, G(y(\xi))\,\mathrm{d}\xi\,. \tag{9}$$

The inputs $u(x)$ and the kernels $K(x,\xi)$ present within the integral terms (9) are supposed to be C^ρ for some $\rho \geq 0$. For instance, we want to allow inputs to be piecewise defined and kernels to be given by, say, cubic splines. It is required that the integral upper bounds $\beta(x) \leq x$ (typically, $\beta(x) = x$) in order to obtain "causal" systems; various lower bounds are allowed (typically $\alpha(x) = x_0$ or $\alpha(x) = x - T$ for some $T > 0$). Some initial values need also be given. Depending on integral lower bounds $\alpha(x)$, the value of $y(x)$ may need to be prescribed on some sufficiently large interval.

In this article, we are concerned with the integration problem by means of a numerical integrator derived from explicit Runge-Kutta methods. Moreover, we focus on the study of "fixed step size" integrators. On the one hand, once such an integrator is designed, it is not difficult to design an adaptive step size integrator following the approach which is classical for ODE — since embedded formulas are available. See [21, sec. II.4]. On the other hand, adaptive step size controllers use the knowledge of the orders of both the principal and the embedded formulas in order to estimate the local error. It is thus important to make sure that the theoretical orders of these formulas correspond to their practical orders — an investigation to be carried out using a "fixed step size" integrator.

The quotes surrounding the qualifier "fixed step size" are due to the fact that step sizes will actually vary during the integration process. Indeed, assuming

some number of steps N is prescribed, one can define a reference step size $h_{\mathrm{r}} = (x_{\mathrm{end}} - x_0)/N$. Assuming moreover that an order p Runge-Kutta method is prescribed, one expects the local error produced by the explicit Runge-Kutta algorithm to be of the order of h_{r}^{p+1} by [21, Theorem 3.4]. Now, if we had to integrate an ODE, it would be sufficient to perform N steps of size h_{r}. This strategy does not apply here because we also want to avoid solving integral equations or, more generally, implicit equations involving integrals.

Avoid Solving Integral Equations Assume that the current point (x_0, y_0) is known. Consider some integral (9) to be evaluated at $x = x_0$. Assume thus that an approximation of $y(\xi)$ is known over the interval $[\alpha(x_0), x_0]$. Since $\beta(x) \leq x$, we have $[\alpha(x_0), \beta(x_0)] \subset [\alpha(x_0), x_0]$ and the integral (9) can be approximated by a mere quadrature. Thus $f(x_0, y_0)$ also can be approximated by quadratures and, given any step size h, the order 1 Euler method (10) permits to compute an approximation of the next point (x_1, y_1)

$$y_1(h) = y_0 + h\, f(x_0, y_0)\,. \tag{10}$$

This is however not true anymore for order $p > 1$ classical Runge-Kutta methods. Consider Runge midpoint formula, summarized by the following Butcher tableau[4] with $s = 2$ stages [21, ch. II.1, Table 1.1]

$$\begin{array}{c|cc} 0 & & \\ c_2 & a_{21} & \\ \hline & b_1 & b_2 \end{array} \quad = \quad \begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}$$

The Runge-Kutta formula [21, II.1, (1.8)] requires s evaluation of the function f of formula (8). These evaluations have the form

$$k_i = f(x_0 + c_i\, h, y_0 + h\,(a_{i,1}\, k_1 + \cdots a_{i,i-1}\, k_{i-1})) \quad (1 \leq i \leq s)$$

Assuming (x_0, y_0) is the current, known, position and the stepsize $h > 0$, we see that negative c_i correspond to an evaluation of f for $x < x_0$ i.e. in the past. *A contrario*, if any c_i is positive (which is the case for all classical tableaux), the evaluation of the formula implies an evaluation in the future which, in the context of IDE, implies to solve an integral equation — or worse. To overcome this issue, we have designed the Butcher tableaux of Fig. 3 with negative c_i only. They were obtained, using the MAPLE computer algebra system, by brute force identification of the coefficients of the Taylor series of the exact solution $y(x_0+h)$ and the ones of the result of the Runge-Kutta formula, denoted $y_1(h)$ in [21, II.1,

[4] Butcher tableaux were introduced by Butcher in [16] to provide a compact description of "Runge-Kutta methods". To each tableau is associated a number of stages (customarily denoted s) and an order (customarily denoted p). The computational cost of a Runge-Kutta method increases with the number of stages. The efficiency increases with the order. The coefficients of the tableaux are denoted c_i (the leftmost column), b_j (the bottom row) and $a_{i,j}$ (the matrix).

(1.8)]. The rightmost tableau has 5 stages since a Gröbner basis computation proved that all 4 stages tableaux of order 4 must have $c_4 = 1$ (a result which is known, at least under some simplifying assumptions — see [21, Theorem 1.6]).

0		
$-\frac{1}{2}$	$-\frac{1}{2}$	
y_1	2	-1
$\hat{y}_1$	1	0

0			
$-\frac{1}{3}$	$-\frac{1}{3}$		
$-\frac{2}{3}$	$-\frac{4}{9}$	$-\frac{2}{9}$	
y_1	$\frac{19}{4}$	-6	$\frac{9}{4}$
$\hat{y}_1$	$\frac{5}{2}$	$-\frac{3}{2}$	0

0				
$-\frac{1}{4}$	$-\frac{1}{4}$			
$-\frac{1}{2}$	0	$-\frac{1}{2}$		
$-\frac{3}{4}$	0	$-\frac{3}{8}$	$-\frac{3}{8}$	
-1	$-\frac{23}{123}$	$\frac{10}{41}$	$-\frac{38}{41}$	$-\frac{16}{123}$
y_1	$\frac{35}{6}$	0	$-\frac{58}{3}$	$\frac{64}{3}$	$-\frac{41}{6}$
$\hat{y}_1$	1	0	$\frac{29}{3}$	$-\frac{52}{3}$	$\frac{23}{3}$

Fig. 3. The leftmost tableau has $s = 2$ stages, order $p = 2$ and an embedded formula of order $\hat{p} = 1$ (Euler). The tableau in the middle has $s = 3$ stages, order $p = 3$ and an embedded formula of order $\hat{p} = 2$. The rightmost tableau has $s = 5$ stages, order $p = 4$ and an embedded formula of order $\hat{p} = 3$. The coefficients c_i of all tableaux (see the leftmost columns) satisfy $-1 \leq c_i \leq 0$ for $1 \leq i \leq s$.

Stability Analysis From a theoretical point of view, the stability of Butcher tableaux can be determined by computing the stability function $R(z)$ of each tableau and establishing that its stability domain — which is the subset of the complex plane such that $|R(z)| < 1$ — is non empty. Some existing computer algebra software are dedicated to this study [38] but we could not take advantage of them by lack of access to Mathematica. Instead, we directly computed $R(z)$ using [22, IV, (2.8)]. We observed that our two first tableaux, for which $p = s$, exhibit the stability function given in [22, IV, (2.12)]. The leftmost tableau has the following stability function, which admits a non empty stability domain:

$$R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} - \frac{z^5}{24}.$$

Experimental evidence of the existence of non empty stability regions for the tableaux of Fig. 3 is provided in Section 4.

Step Size Control Runge-Kutta methods with $c_i < 0$ have however a drawback when x_0 is the initial value or is on the border of a piecewise defined domain, since the integrator will try to estimate the current derivative of the integral curve on the right hand piece of the domain from derivatives evaluated on the left hand piece. This drawback is certainly a feature for integral terms (by design of the equations). But the terms which lie outside integrals should be evaluated on the right hand piece of the integration domain.

To achieve this goal, our strategy consists in starting with a single Euler step, using a very small step size h_0, then switch to some prescribed more efficient

Runge-Kutta method of Fig. 3 and double the step size at each iteration until the reference step size h_r is reached. Precisely, assume we want to apply some Runge-Kutta method of order $p > 1$. We expect a local error of order h_r^{p+1}. This local error can also be obtained by an Euler step with step size h_0 such that $h_0^2 = h_\mathrm{r}^{p+1}$ i.e. such that $h_0 = h_\mathrm{r}^{(p+1)/2}$. Let now k be an integer such that $h_0 \simeq h_\mathrm{r}/2^k$. Solving, one gets

$$k = \left\lceil -\log_2 \left(h_\mathrm{r}^{\frac{p-1}{2}} \right) \right\rceil, \qquad h_0 = \frac{h_\mathrm{r}}{2^k}.$$

Let us assume we are starting the integration with x_0 precisely on the border between two pieces of the integration interval or at its beginning. The first Euler step with step size h_0 does not involve any negative c_i: the terms depending on y and lying outside integrals are evaluated over the border, which may be considered as part of the right hand piece. The second iteration starts at $x_0 + h_0$. Since the coefficients c_i of Fig. 3 satisfy $0 \geq c_i \geq -1$, this step can be performed using the order p Runge-Kutta method, with step size h_0: all terms depending on y and lying outside integrals are thus evaluated within the right hand piece. The third iteration starts at $x_0 + 2\,h_0$. This step can be performed using the order p Runge-Kutta method, with step size $h = 2\,h_0$. Continue likewise, doubling the step size at each iteration. At the iteration $k + 2$, the reference step $h = h_\mathrm{r}$ is reached (see below) and the integrator may continue with this fixed step size.

Step number	Step size	Method
1	$h_0 = h_\mathrm{r}/2^k$	Euler
2	h_0	Order p RK
3	$2\,h_0$	Order p RK
	$\vdots$	
$k+2$	$2^k\,h_0 = h_\mathrm{r}$	Order p RK

Evaluating Quadratures In order to evaluate quadratures at any x, it is necessary to be able to evaluate the dependent variable y at any $\xi \in [x_0, x]$.

For this, the whole sequence of points (x_k, y_k) computed by the numerical integrator is recorded as well as the value $f_k = f(x_k, y_k)$ (the derivative of y) whenever it is available. Two methods are implemented for estimating $y(\xi)$:

1. by evaluating the interpolation polynomial defined by a set of points surrounding ξ (the optimal number of points depends on the order of the Butcher tableau), using Newton's divided differences;
2. by evaluating the interpolation polynomial defined by Hermite interpolation i.e. over a dense output of the integrator. See [21, ch. II.6].

For quadratures, since the orders of our tableaux do not exceed 4, we use basic integration schemes i.e. Newton and Simpson order 4 formulas, with step size equal to the reference step size h_r.

4 Implementation

Our numerical integrator is implemented within an open source C library (about 4000 lines plus 3200 lines for the test suite of version 2.1) available at [5]. It compiles over Linux platforms. It is endowed with a MAPLE library which considerably simplifies the C code generation from mathematical systems.

The C code can be compiled using floating point numbers of various sizes (simple, double, long double and quadruple precision). Its main functionalities are a fixed step size numerical integrator for IDE systems and a function which seeks the best fitting parameters of an IDE system w.r.t. experimental data. This function is mostly a call to the GSL implementation of the Levenberg-Marquardt algorithm, which relies on our numerical integrator in order to compute errors.

4.1 Data Types

Here is a quick review of the main data types. For a better flexibility, most of them are parametrized by functions.

The library has been designed to apply over a piecewise defined integration interval. Pieces may arise from many different sources: inputs may be piecewise defined, delayed evaluations such as $y(x - T)$ may occur from differentiated integral terms ... The boundaries between the pieces of the integration interval are called *critical points*.

A specific data type permits to describe the possibly many different inputs $u(x)$ of the IDE system to be integrated. An input is defined by a name, an evaluation function and a function which permits to enlarge the set of the model critical points with the ones which are due to the input.

A specific data type is dedicated to the model parameters. A parameter is defined by a name, a floating point value, a function which permits to enlarge the set of the model critical points with the ones which are related to the parameter, and two functions providing a transformation and its reciprocal before performing nonlinear fitting methods (an example of such a useful pair of transformations is the pair log / exp to keep positive small parameters which must remain positive).

A specific data type describes the problem i.e. the IDE system to be integrated. A problem is defined by a dimension n, an integration interval $[x_0, x_{\text{end}}]$, an array of n initial value functions (in the general case, the numerical integration of an IDE requires the knowledge of the dependent variable y over an interval, not only a single value at x_0), an array of inputs, an array of parameters and a function *fcn* for evaluating the right hand sides of the IDE equations. A field of the problem data structure contains a description of the problem critical points.

The integral terms (9) occuring in the right hand sides of the IDE equations are described in a separate array of the problem data structure. This permits to evaluate them before calling *fcn* in order to speed up the integrator as follows. Recall that, at each integration step, the m (say) integral terms have to be evaluated s (the number of stages) times. Thus: 1) by grouping the $m \times s$

quadrature evaluations in the code, it is much easier to compute them in parallel using OpenMP facilities; 2) in some cases, the s evaluations of a given integral term can be computed almost at the cost of a single evaluation, by updating the current result from one stage to the following one.

A last data type contains the whole data needed by the integration process (it is called the "history"). It contains the sequence of all points computed so far (which is the actual history), the problem, the Butcher tableau to be used and a few other fields of minor importance.

4.2 Usefulness of the Computer Algebra Package

A MAPLE package, called `Blineide` and shipped with the C library, permits to handle IDE problems given by mathematical formulas. It permits either to directly perform computations from MAPLE or to generate C code to help programmers who want to work at C level.

Beyond the obvious simplification provided to the user, the idea of generating C code from a computer algebra software provides two important enhancements which are not yet implemented: 1) it should permit to detect linear (algebraic?) dependencies between the integral terms occuring in the IDE model and use this information to reduce the computation cost; 2) it might permit a symbolic study of the location of critical points for a better reliability of the integrator.

4.3 Tests

Checking Convergence Towards Exact Solution Some tests are designed to check that the numerical integrator converges toward the true solution of a given IDE system, with the expected experimental order. An example of such an IDE is the following one, which admits $y(x) = \cos(x)$ as a solution:

$$\begin{aligned} &\dot{y}(x) = \sin(x) - y^2(x) + 4 \int_0^x (x-\xi)^2 \, \sin(\xi)\, y(\xi)\, \mathrm{d}\xi - x^2 + 1\,, \\ &y(0) = 1\,. \end{aligned}$$

In order to test the experimental order of the numerical integrator over a given example, the test function computes the relative error produced with 2^k integration steps, for many different values of k. The experimental order is then estimated, by linear least squares, as the slope a of the following equation:

$$k\,a + b = -\log_2(\mathrm{relerr})\,. \tag{11}$$

Other tests check the behaviour of the numerical integrator using various inputs and kernels, including kernels defined by cubic splines.

Checking Experimental Orders We checked our integrator over Volterra-Kostitzin model (1) and the compartmental IDE model (3). In the case of the Volterra-Kostitzin model, we estimated the practical order of the integrator when

used with the Butcher tableaux of Fig. 3. In particular, we addressed the case of non smooth kernels in integrals (see the curves of Fig. 4) and non smooth inputs (curves not shown).

On the left hand picture of Fig. 4, the kernel is a cubic spline (i.e. a C^2 curve). On the right hand picture, the kernel is a smooth curve (on the two pictures, the mathematical problem to be solved is thus not the same). In each picture, there is one curve per Butcher tableau of Fig. 3. Each curve was obtained as follows: a first integration was performed with 2^{15} steps. Its result was then considered as a reference value and compared with the result of other integrations with $2^8, 2^9, \ldots, 2^{14}$ steps, giving 7 points hence a curve, which should be a straight line (see formula (11)). Its slope is a numerical estimate of the order of the numerical integrator. In the case of a C^2 kernel, the integrator has order 2 when used with an order 2 tableau; and a non reliable order close to 2 when used with order 3 and 4 tableaux. In the case of a smooth kernel, the integrator has the same order as the tableau with which it is used (the curve for order 4 is slightly irregular because the order of the quadrature formula does not match the one of the Butcher tableau).

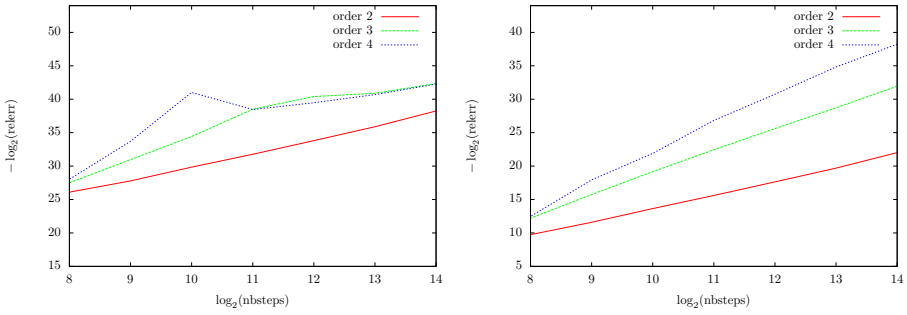

Fig. 4. Experimental evaluation of the order of the IDE numerical integrator over Volterra-Kostitzin model (1), with an integral lower bound equal to zero.

Nonlinear Fit A test solves the fitting problem addressed by Kostitzin over data obtained on a population of *staphylococcus*, obtaining a much better result than [26, page 72] which is to be expected since Kostitzin estimated parameters using his mathematical skills, without any computer! See Figure 5.

Conclusion

We have presented and discussed a symbolic method for computing the IO equation of a given IDE system which is likely to apply over an important class of IDE

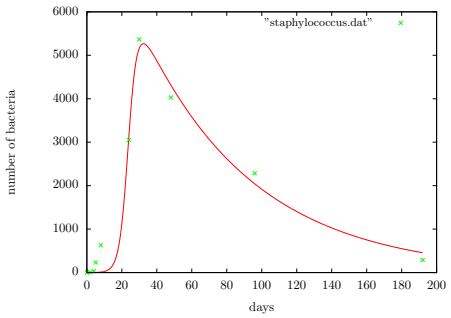

Fig. 5. Best fitting curve for (1) with the trivial kernel $K(x, \xi) = 1$ and an integral lower bound equal to zero, against the *staphylococcus* population reported in [26, page 72]. Optimal parameters are $\varepsilon = 3.97 \times 10^{-1}$, $\lambda = 6.56 \times 10^{-5}$ and $\mu = 1.02 \times 10^{-6}$.

models, together with an open source library dedicated to the numerical integration of such systems, endowed with a new MAPLE package. This library does not only integrate IDE systems but provides also parameter estimation facilities. It seems to have no available equivalent. Its existence is of major importance for promoting IDE modeling.

However, these very promising results raise in turn many fascinating challenges, both theoretical and practical. Indeed, what about: a complete algorithm for computing IO equations? an IDE analogue of the "input-output ideal" method? a sound theory for critical points? implicit numerical integrators? These issues will be addressed in future papers.

Acknowledgement. This work has been supported by the bilateral project ANR-17-CE40-0036 and DFG-391322026 SYMBIONT.

References

1. Bächler, T., Gerdt, V., Lange-Hegermann, M., Robertz, D.: Algorithmic Thomas Decomposition of Algebraic and Differential Systems. Journal of Symbolic Computation 47(10), 1233–1266 (2012)
2. Bavula, V.V.: The algebra of integro-differential operators on a polynomial algebra. J. Lond. Math. Soc. 83(2), 517–543 (2011)
3. Bavula, V.V.: The algebra of integro-differential operators on an affine line and its modules. Journal of Pure and Applied Algebra 17(3), 495–529 (2013)
4. Bavula, V.V.: The algebra of polynomial integro-differential operators is a holonomic bimodule over the subalgebra of polynomial differential operators. Algebras and Representation Theory 17(1), 275–288 (2014)
5. Boulier, F., al.: BLINEIDE. `http://cristal.univ-lille.fr/~boulier/BLINEIDE`
6. Boulier, F., Lemaire, F.: A Normal Form Algorithm for Regular Differential Chains. Mathematics in Computer Science 4(2), 185–201 (2010), `http://dx.doi.org/10.1007/s11786-010-0060-3`

7. Boulier, F., Cheb-Terrab, E.: DifferentialAlgebra. Package of MapleSoft MAPLE standard library since MAPLE 14 (2008)
8. Boulier, F., Korporal, A., Lemaire, F., Perruquetti, W., Poteaux, A., Ushirobira, R.: An Algorithm for Converting Nonlinear Differential Equations to Integral Equations with an Application to Parameter Estimation from Noisy Data. In: LNCS 8660: Proceedings of Computer Algebra and Scientific Computing (CASC) 2014. pp. 28–43. Warsaw, Poland (2014)
9. Boulier, F., Lallemand, J., Lemaire, F., Regensburger, G., Rosenkranz, M.: Additive normal forms and integration of differential fractions. Journal of Symbolic Computation 77, 16–38 (2016)
10. Boulier, F., Lazard, D., Ollivier, F., Petitot, M.: Representation for the radical of a finitely generated differential ideal. In: ISSAC'95: Proceedings of the 1995 international symposium on Symbolic and algebraic computation. pp. 158–166. ACM Press, New York, NY, USA (1995), http://hal.archives-ouvertes.fr/hal-00138020
11. Boulier, F., Lazard, D., Ollivier, F., Petitot, M.: Computing representations for radicals of finitely generated differential ideals. Applicable Algebra in Engineering, Communication and Computing 20(1), 73–121 (2009), http://dx.doi.org/10.1007/s00200-009-0091-7, (1997 Techrep. IT306 of the LIFL)
12. Boulier, F., Lemaire, F., Moreno Maza, M.: Computing differential characteristic sets by change of ordering. Journal of Symbolic Computation 45(1), 124–149 (2010), doi:10.1016/j.jsc.2009.09.04
13. Boulier, F., Lemaire, F., Moreno Maza, M., Poteaux, A.: An Equivalence Theorem For Regular Differential Chains. Journal of Symbolic Computation (2018), (to appear)
14. Boulier, F., Lemaire, F., Rosenkranz, M., Ushirobira, R., Verdière, N.: On Symbolic Approaches to Integro-Differential Equations. Advances in Delays and Dynamics, Springer (2017), preprint available at https://hal.archives-ouvertes.fr/hal-01367138
15. Brunner, H., van der Hoeven, P.J.: The numerical solution of Volterra equations. North-Holland, Amsterdam (1986)
16. Butcher, J.C.: On Runge-Kutta Processes of High Order. J. Austral. Math. Soc. IV, Part 2, 179–194 (1964)
17. Denis-Vidal, L., Joly-Blanchard, G., Noiret, C.: System identifiability (symbolic computation) and parameter estimation (numerical computation). In: Numerical Algorithms. vol. 34, pp. 282–292 (2003)
18. Feldstein, A., Sopka, J.R.: Numerical Methods for Nonlinear Volterra Integro-Differential Equations. SIAM J. Numer. Anal. 11(4), 826–846 (1974)
19. Gao, X., Guo, L.: Constructions of Free Commutative Integro-Differential Algebras. In: Barkatou, M., Cluzeau, T., Regensburger, G., Rosenkranz, M. (eds.) Algebraic and Algorithmic Aspects of Differential and Integral Operators. LNCS, vol. 8372, pp. 1–22 (2014)
20. Guo, L., Regensburger, G., Rosenkranz, M.: On integro-differential algebras. Journal of Pure and Applied Algebra 218(3), 456–473 (2014)
21. Hairer, E., Norsett, S.P., Wanner, G.: Solving ordinary differential equations I. Nonstiff problems, Springer Series in Computational Mathematics, vol. 8. Springer–Verlag, New York, 2nd edn. (1993)
22. Hairer, E., Wanner, G.: Solving ordinary differential equations II. Stiff and Differential–Algebraic Problems, Springer Series in Computational Mathematics, vol. 14. Springer–Verlag, New York, 2nd edn. (1996)

23. Jerri, A.J.: Introduction to Integral Equations with Applications, Monographs and Textbooks in Pure and Applied Mathematics, vol. 93. Marcel Dekker Inc. (1985)
24. Keener, J., Sneyd, J.: Mathematical Physiology I: Cellular Physiology, Interdisciplinary Applied Mathematics, vol. 8/I. Springer Verlag, second edn. (2010)
25. Kolchin, E.R.: Differential Algebra and Algebraic Groups. Academic Press, New York (1973)
26. Kostitzin, V.A.: Biologie Mathématique. Armand Colin (1937), (with a foreword by Vito Volterra)
27. Ljung, L., Glad, S.T.: On global identifiability for arbitrary model parametrisations. Automatica 30, 265–276 (1994)
28. Mansfield, E.L.: Differential Gröbner Bases. Ph.D. thesis, University of Sydney, Australia (1991)
29. Moulay, D., Verdière, N., Denis-Vidal, L.: Identifiability of parameters in an epidemiologic model modeling the transmission of the Chikungunya. In: In Proceedings of the 9ème Conférence Internationale de Modélisation, Optimisation et SIMulation (2012)
30. Ollivier, F.: Le problème de l'identifiabilité structurelle globale : approche théorique, méthodes effectives et bornes de complexité. Ph.D. thesis, École Polytechnique, Palaiseau, France (1990)
31. Paulsson, J., Elf, J.: Stochastic Modeling of Intracellular Kinetics. In: Szallasi, Z., Stelling, J., Periwal, V. (eds.) System Modeling in Cellular Biology: From Concepts to Nuts and Bolts. pp. 149–175. Cambridge, Massachussets: The MIT Press (2006)
32. Pavé, A.: Modeling living systems, from cell to ecosystem. ISTE/Wiley (2012)
33. Quadrat, A., Regensburger, G.: Polynomial solutions and annihilators of ordinary integro-differential operators. IFAC Proceedings Volumes 46(2), 308–313 (2013)
34. Reid, G.J., Wittkopf, A.D., Boulton, A.: Reduction of systems of nonlinear partial differential equations to simplified involutive forms. European Journal of Applied Math. 7(6), 635–666 (1996)
35. Ritt, J.F.: Differential Algebra, American Mathematical Society Colloquium Publications, vol. 33. American Mathematical Society, New York (1950)
36. Rosenkranz, M., Regensburger, G.: Integro-Differential Polynomials and Operators. In: Jeffrey, D. (ed.) ISSAC'08: Proceedings of the 2008 International Symposium on Symbolic and Algebraic Computation. ACM Press (2008)
37. Shampine, L.F., Thompson, S.: Solving DDEs in MATLAB. Applied Numerical Mathematics 37, 441–458 (2001)
38. Sofroniou, M.: Order Stars and Linear Stability Theory. Journal of Sumbolic Computation 21, 101–131 (1996)
39. Verdière, N., Denis-Vidal, L., Joly-Blanchard, G.: A new method for estimating derivatives based on a distribution approach. Numerical Algorithms 61, 163–186 (2012)
40. Verdière, N., Denis-Vidal, L., Joly-Blanchard, G., Domurado, D.: Identifiability and Estimation of Pharmacokinetic Parameters for the Ligands of the Macrophage Mannose Receptor. Int. J. Appl. Math. Comput. Sci. 15(4), 517–526 (2005)
41. Wikipedia: Delay Differential Equations. `https://en.wikipedia.org/wiki/Delay_differential_equation`
42. Zhu, S.: Modeling, identifiability analysis and parameter estimation of a spatial-transmission model of chikungunya in a spatially continuous domain. Ph.D. thesis, Université de Technologie de Compiègne, Compiègne, France (2017)